Supplementary Materials #695

A. Source Coding

Source Code: A source code C is a mapping from the range of a random variable or a set of random variables to finite length strings of symbols from a K-ary alphabet.

Expected length of a source code denoted by $L(C)$ is given as follows: $L(C) = \sum_{x \in \mathcal{X}} p(x) l(x)$, where $l(x)$ is the length of codeword $c(x)$ for a symbol $x \in \mathcal{X}$, and $p(x)$ is the probability of the symbol.

Intuitively, a good code should preserve the information content of an outcome. Since information content depends on the probability of the outcome (it is higher if probability is lower, or equivalently if the outcome is very uncertain), a good codeword will use fewer bits to encode a certain or high probability outcome and more bits to encode a low probability outcome. Thus, we expect that the smallest expected code length should be related to the average uncertainty of the random variable, i.e., the entropy.

Source coding theorem states that entropy is the fundamental limit of data compression; i.e., $\forall C : L(C) \geq H(X)$. Instead of encoding individual symbols, we can also encode blocks of symbols together. A length n block code encodes n length strings of symbols together and is denoted by $C\left(x_1, \ldots, x_n\right) =: C\left(x^n\right)$.

Proof. Consider the optimization problem:

$$\min_{l(x)} \sum_{x \in \mathcal{X}} p(x) l(x) \quad \text{such that} \quad \sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$$

The above finds the shortest possible code length subject to satisfying the Kraft inequality. If we relax the the codelengths to be non-integer, then we can obtain a lower bound. To do this, the Lagrangian is:

$$\mathcal{L} = \sum_{x \in \mathcal{X}} p(x) l(x) + \lambda \left(\sum_{x \in \mathcal{X}} 2^{-l(x)} - 1 \right)$$

Taking derivatives with respect to $l(x)$ and λ and setting to 0, leading to:

$$\begin{aligned} p(x) + \ln 2 \lambda 2^{-l(x)} &= 0 \\ \sum_{x \in \mathcal{X}} 2^{-l(x)} - 1 &= 0 \end{aligned}$$

Solving this for $l(x)$ leads to $l(x) = \log \frac{1}{p(x)}$, which can be verified by direct substitution. This proves the lower bound. □

Theoretical analysis can be seen in (Shannon, 1948).

B. Maximum Data Rate

We first introduce the Nyquist ISI criterion:

Proposition 1 (Nyquist ISI criterion)**.** *If we denote the channel impulse response as $h(t)$, then the condition for an ISI-free response can be expressed as:*

$$h\left(nT_s\right) = \begin{cases} 1; & n = 0 \\ 0; & n \neq 0 \end{cases}$$

for all integers n, where T_s is the symbol period. The Nyquist ISI criterion says that this is equivalent to:

$$\frac{1}{T_s} \sum_{k=-\infty}^{+\infty} H\left(f - \frac{k}{T_s}\right) = 1 \quad \forall f$$

where $H(f)$ is the Fourier transform of $h(t)$.

We may now state the Nyquist ISI criterion for distortionless baseband transmission in the absence of noise: The frequency function $H(f)$ eliminates intersymbol interference for samples taken at interval T_s provide that it satisfies Equation 1.

The simplest way of satisfing Equation 1 is to specify the frequency function $H(f)$ to be in the form of a rectangular function, as showing by

$$\begin{aligned} H(f) &= \begin{cases} \frac{1}{2\mathbb{W}}, & -W < f < \mathbb{W} \\ 0, & |f| > \mathbb{W} \end{cases} \\ &= \frac{1}{2W} \operatorname{rect}\left(\frac{f}{2W}\right) \end{aligned}$$

where $\operatorname{rect}(f)$ stands for a rectangular function of unit amplitude and unit support centered on $f = 0$, and the overall system bandwidth W is definded by

$$W = \frac{1}{2T_s} = \frac{R_b}{2}$$

The special value of the bit rate $R_b = 2W$ is called the Nyquist rate, and W is itself called the Nyquist bandwidth.

The key here is that we have restricted ourselves to binary transmission and are limited to $2W$ bits/s no matter how much we increase the signal-to-noise ratio. The way to attain a higher R_b value is to replace the binary transmission system with a multilevel system, often termed an K-ary transmission system, with $K > 2$. An K-ary channel can pass $2Wlog_2K$ bits/s with an acceptable error rate.

Thus, we conclude that the bit rate $R_b \leq R_{max} = 2Wlog_2K$.

C. The Information Bottleneck Method

C.1. Background

The information bottleneck method provides a principled way to extract information that is present in one variable that is relevant for predicting another variable. Consider X and Y respectively as the input source and target, and let Z be an internal representation, i.e., a stochastic encoding, of any hidden layer of the network, defined by a parametric encoder $p(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$. The goal is to learn an encoding that is maximally informative about the target Y, which is measured by the mutual information between Y and Z, where

$$I(Z,Y;\boldsymbol{\theta}) = \int p(z,y|\boldsymbol{\theta}) \log \frac{p(z,y|\boldsymbol{\theta})}{p(z|\boldsymbol{\theta})p(y|\boldsymbol{\theta})} dxdy \tag{1}$$

Notice that taking the identity encoding always ensures a maximally informative representation if only with the above objective, but it is not a useful representation obviously. It is evident to constrain on encoding's complexity if we want the best representation, i.e., Z. (Tishby and Zaslavsky, 2015) proposed the information bottleneck that expresses the trade-off between the mutual information measures $I(X,Z)$ and $I(Z,Y)$. This suggests the objective:

$$\max_{\boldsymbol{\theta}} \; I(Z,Y;\boldsymbol{\theta}) \quad s.t.\; I(X,Z;\boldsymbol{\theta}) \leq I_c \tag{2}$$

where I_c is the information constraint. Equivalently, with the introduction of a Lagrange multiplier β we can maximize the objective function:

$$L(\boldsymbol{\theta}) = I(Z,Y;\boldsymbol{\theta}) - \beta I(X,Z;\boldsymbol{\theta}) \tag{3}$$

where β controls the trade-off. Intuitively, the first term encourages Z to be predictive of Y ; the second term encourages Z to "forget" X. Essentially it forces Z to act like a minimal sufficient statistic of X for predicting Y.

C.2. Why IMAC works?

We discuss why information bottleneck works in multi-agent communication.

We first introduce minimal sufficient statistics: a transformation $T(X)$ of the data X is a minimal sufficient statistic if $T(X) \in \arg\min_S I(X, S(X))$, where $S(X)$ is s.t. $I(\theta, S(X)) = \max_{T'} I(\theta, T'(X))$

Information bottleneck principle generalizes the notion of minimal sufficient statistics and suggests using a summary of the data $T(X)$ that has least mutual information with the data X while preserving some amount of information about an auxiliary variable Y.

According to (Shamir et al., 2010), from a learning perspective, we discuss the role of $I(X;T)$, the compression or minimality term in information bottleneck, as a regularizer when maximizing $I(Y;T)$.

Reinforcement learning learns a optimal action given a state. If we know the optimal action in advance, like imitation learning, then we would maximize the mutual information between the state and its corresponding optimal action, which is a straightforward application of supervised learning. Here, X represents the states, T represents the messages, Y represents the actions. Note that without regularization, $I(Y;T)$ can be maximized by setting $T = X$. However, $p(x|y)$ cannot be estimated efficiently from a sample of a reasonable size; It means that more samples are needed in reinforcement learning. In another words, methods with regularization on $I(X;T)$, e.g., IMAC, can accelerate convergence.

D. Experimental Details and Results

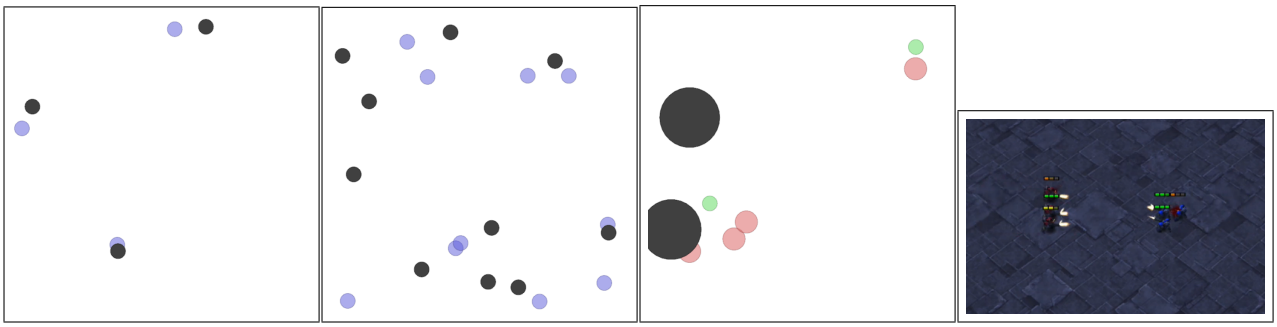

Figure 1: Illustration of the experimental environments. (a) Cooperative navigation of 3 agents; (b) Cooperative navigation of 10 agents; (c) Predator and Prey; (d) 3m in StarCraft II.

D.1. Cooperative navigation

In this environment, agents must cooperate through physical actions to reach a set of landmarks. Agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the proximity of any agent to each landmark. In other words, the agents have to 'cover' all of the landmarks. Further, the agents occupy significant physical space and are penalized when colliding with each other. Our agents learn to infer the landmark they must cover, and move there while avoiding other agents.

Training details We set the number of agents as 3,5,10 respectively. We use MLP with hidden layer size of 64 as basic module as before communication model, after communication model. We use the ADAM as optimizer with learning rate of 0.01. Since the environment of cooperative navigation does not send "terminal/done" to agents, we set each episode with a maximal steps of 25. The reward of each agent is identical, which equals to the sum of distances between agents to their nearest landmark. It means that agents are required not only to approach its nearest landmark, but also share information with each other for a common goal. We use the same hyper-parameter as MADDPG of openAI's version.

Table 1 shows that MADDPG without communication tends to use high-entropy messages, while IMAC can convey low-entropy messages. Combined with the performance in Figure 4, we can see that under limited-bandwidth constraint, IMAC learns informative communication protocols.

D.2. Predator and prey

In this variant of the classic predator-prey game, some slower cooperating agents must chase some faster adversaries around a randomly generated environment with some large landmarks impeding the way. Each time the cooperative agents collide with some adversaries, the agents are rewarded while the adversary is penalized. Agents observe the relative positions and velocities of the agents, and the positions of the landmarks.

Training details We set the number of predators as 4, the number of preys as 2, and the number of landmarks as 2. We use the same architecture and hyper-parameter as configuration in cooperative navigation. We trained our agents by self-play for 100,000 episodes and then evaluate performance by cross-comparing between IMAC and the baselines. We average the episode rewards across 1000 rounds (episodes) as scores.

$sigma^2$	None	1	5	10
$H(M_i) = \frac{1}{2}\log(2\pi e\sigma^2)$	-	1.419	2.223	2.570
maddpg w/ com	3.480+-0.042	1.530+-0.038	3.147+-0.088	3.891+-0.059
IMAC train w/ bw=1	0.244+-0.028	-	-	-
IMAC train w/ bw=5	2.227+-0.002	1.383+-0.003	-	-
IMAC train w/ bw=10	2.763+-0.215	1.695+-0.044	3.017+-0.026	-

Table 1: Entropy of messages in different limited bandwidths (number in cell represents $H(M_i) = \frac{1}{2}\log(2\pi e\sigma^2)$, which is calculated based on running variance).

D.3. Predator and prey

In this variant of the classic predator-prey game, some slower cooperating agents must chase some faster adversaries around a randomly generated environment with some large landmarks impeding the way. Each time the cooperative agents collide with some adversaries, the agents are rewarded while the adversary is penalized. Agents observe the relative positions and velocities of the agents, and the positions of the landmarks.

Training details We set the number of predators as 4, the number of preys as 2, and the number of landmarks as 2. We use the same architecture and hyper-parameter as configuration in cooperative navigation. We trained our agents by self-play for 100,000 episodes and then evaluate performance by cross-comparing between IMAC and the baselines. We average the episode rewards across 1000 rounds (episodes) as scores.

D.4. StarCraft II

We use the StarCraft Multi-Agent Challenge (SMAC) environment (Samvelyan et al., 2019). SMAC uses the StarCraft II Learning Environment to introduce a cooperative MARL environment, which focus solely on micromanagement. Each unit is controlled by an independent agent that conditions only on local observations restricted to a limited field of view centred on that unit.

We choose two scenarios: 3m and 8m as our testbeds, where a group of marines need to defeat an enemy marines. The feature vector observed by each agent contains the following attributes for both allied and enemy units within the sight range: distance, relative x, relative y, health, shield, and unit type. The discrete set of actions which agents are allowed to take consists of move in four directions, attack an enemy, stop and no-op. Dead agents can only take no-op action while live agents cannot. Some positive (negative) rewards after having enemy (allied) units killed and/or a positive (negative) bonus for winning (losing) the battle.

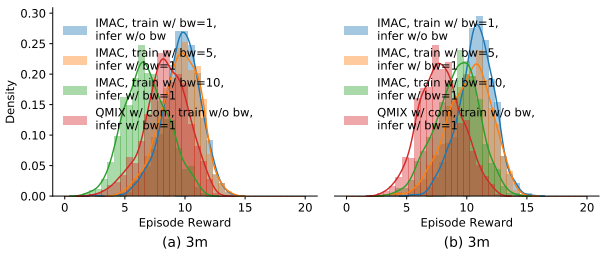

Figure 2: Density plot of episode reward per agent during the execution stage. (a), (b) Reward distribution of IMAC trained with different prior distributions against MADDPG with communication under the same bandwidth constraint in 3m and 8m respectively. "bw=δ" means in the implementation of the limited bandwidth constraint, the variance $\mathbf{\Sigma}$ of Gaussian distribution is δ.

Training details We use QMIX architecture (Rashid et al., 2018) as our algorithm backbone, where a DRQN with a recurrent layer composed of a 64-dimensional GRU and a fully-connected layer before and after. We extend QMIX with communication module. Each agent can receive other's messages from previous timestep. We use the RMSProp optimizer with learning rate = 0.0005 $\alpha = 0.99$, and $\epsilon = 0.00001$. The replay buffer contains the most recent 5000 episodes.We sample batches of 120 episodes uniformly from the replay buffer, and train on fully unrolled episodes, performing a single gradient descent step after every episode. The target networks are updated after every 200 training episodes.

Results Figure 2 shows the density plot of episode reward per agent during the execution stage. We first respectively train IMAC with different prior distributions $z(M_i)$ of $N(0, 1)$, $N(0, 5)$, and $N(0, 10)$, to satisfy a limited bandwidth with the variance of 1. In the execution stage, we constrain these algorithms into different bandwidths. As depicted in Figure 2 (a) and (b), QMIX+IMAC with different prior distributions outperform QMIX with communication in the limited bandwidth environment. Results here demonstrate that IMAC discards useless information without impairment on performance.

References

T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

O. Shamir, S. Sabato, and N. Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015.